Kickstart 2012

Day 1

Intro and Basics

Stephanie Rogers and Amy Pavel

Welcome ☺

- Who are we?

What is Computer Science?

- Problem Solving
- Building things
- CS is everywhere
  - Internet
  - Phone/Web Applications
  - Vehicles
  - Genetics
  - And more!

What is Computer Science? (cont)

- Programming
  - Art and science of constructing artifacts that perform computations
  - Programming languages

What is Kickstart?

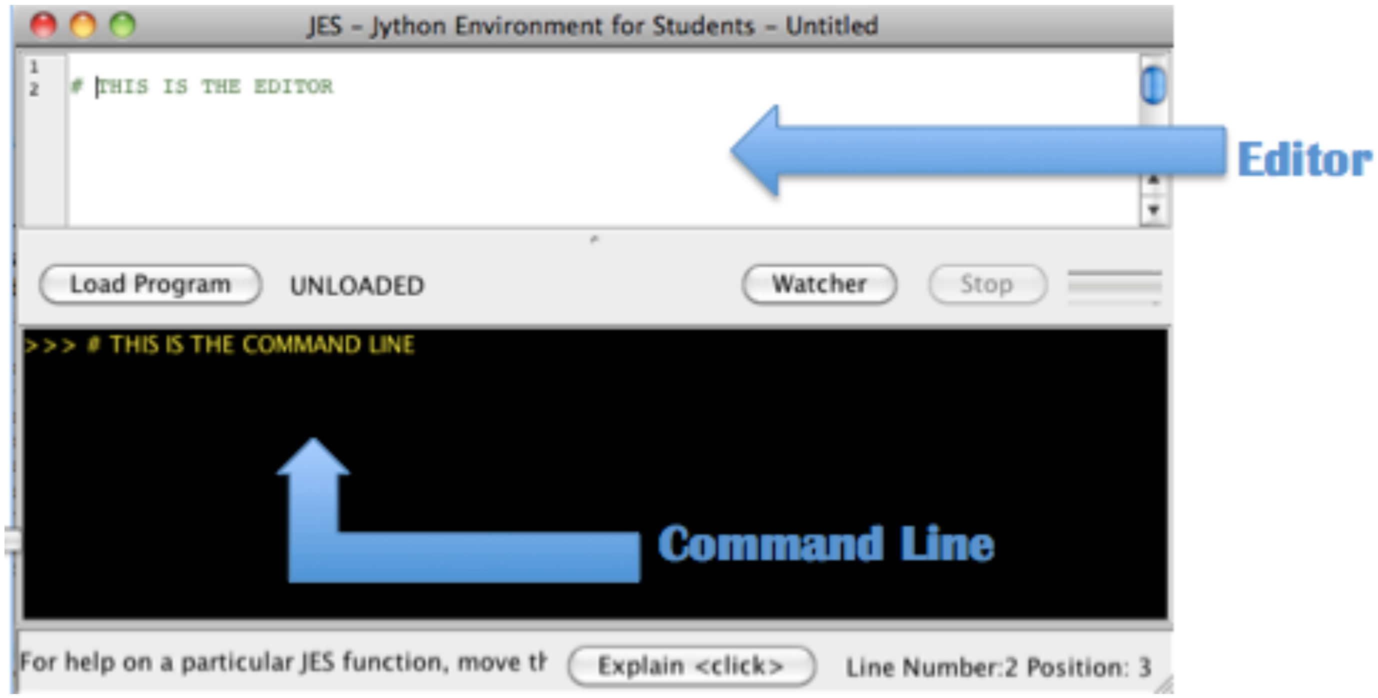
- Not 1's and 0's
- Implementing programs
- Producing a tangible result!
  - **PICTURES**
- An Intro to Jython

Programming Languages

- Communication with computers
- Different encodings of instructions for machines
- The language we are using: Jython
  - Jython is Python!
  - *Java-based Python*
- Ice Breaker – Partner in common (talent)
- Logins!

Environment - JES

- **J**ython **E**nvironment for **S**tudents
- Programming area: the editor, writing programs
- Command area: Entering commands


JES - Jython Environment for Students - Untitled
Editor
Load Program
UNLOADED
Watcher
Stop
Command Line
Explain <click>
Line Number:2 Position: 3

```
# THIS IS THE EDITOR
```

```
>>> # THIS IS THE COMMAND LINE
```

For help on a particular JES function, move th

Meet Jython - Data

Data: stuff we manipulate

➢ integers:	2 -1 13
➢ strings:	“hello world”
➢ booleans:	true, false
➢ lists	[1, 2, 3]
➢ More later	

```
>>> 2
2
>>> “hello world”
‘hello world’
```

Meet Jython - Functions

Functions: rules for manipulate data

➢ Primitive expressions:	+, -, *, /, …
➢ Built-in functions:	sum, abs, …
➢ Self-defined function:	def square(x): …

Can take any number of arguments

Meet Jython - Expressions

Expressions

- Combining functions with data
- Jython evaluates these expressions for you

```
>>> 2+3
?
>>> sum(2, 3)
?
>>> abs(-2)
?
>>> print('hello world')
?
```

Calling functions

- **Remember**
  - Functions: rules for manipulating data
  - Can take any number of arguments

```
>>> x = sum(4, 3)
>>> y = abs(-9)
>>> max(x, y)
9

>>> Can we do all this in one line?
```

Calling functions

- **Remember**
  - Functions: rules for manipulating data
  - Can take any number of arguments

```
>>> x = sum(4, 3)
>>> y = abs(-9)
>>> max(x, y)
9
```

>>> Can we do all this in one line?

Nesting

Calling functions

- **Remember**
  - Functions: rules for manipulating data
  - Can take any number of arguments

```
>>> x = sum(4, 3)
>>> y = abs(-9)
>>> max(x, y)
9

>>> max(  sum(4, 3)   ,   abs(-9)   )
?
```

Meet Jython - Numbers

+, -, *, /, %, >, >=, ==, !=, <, <=

```
>>> 2 + 3
5
>>> (5 * 8) + 2
42
>>> 40 / 5
8.0
>>> 11 % 3
2
```

```
>>> 4 > 3
True
>>> 6 <= 5
False
>>> 6 == (3+3)
True
>>> 6 != 5
True
```

Meet Jython – Logic

Booleans: True, False

Logical operators: and, or, not, >, >= …

and

```
>>> (4 > 3) and (4 < 5)
True
>>> True and False
False
>>> True and True
True
```

or

```
>>> (4 > 3) or (4 > 5)
False
>>> False or False
False
>>> True or False
True
```

Meet Jython - Assignment

Variables

Name our data and functions for use later

```
>>> x = 3
>>> print x + 1
4
```

Meet Jython - Strings

Indexing + concatenation

```
>>> "hi " + "stephanie"
??

>>> print("hello, world")
??

>>> name = "stephanie"
>>> name[0]
??
```

Meet Jython - Lists

Indexing & concatenation

```
>>> [1, 2, 3, 4]
?

>>> [1, 2] + [3, 4]
?

>>> alphabet = [‘a’, ‘b’, ‘c’]
>>> alphabet[2]
??
```

Administrivia

- Website: inst.eecs.berkeley.edu/~cs98-tr

- Lab Structure
  - Lecture
  - Interactive practice - labs
  - Projects

- Send us your pictures daily!

Try it yourself

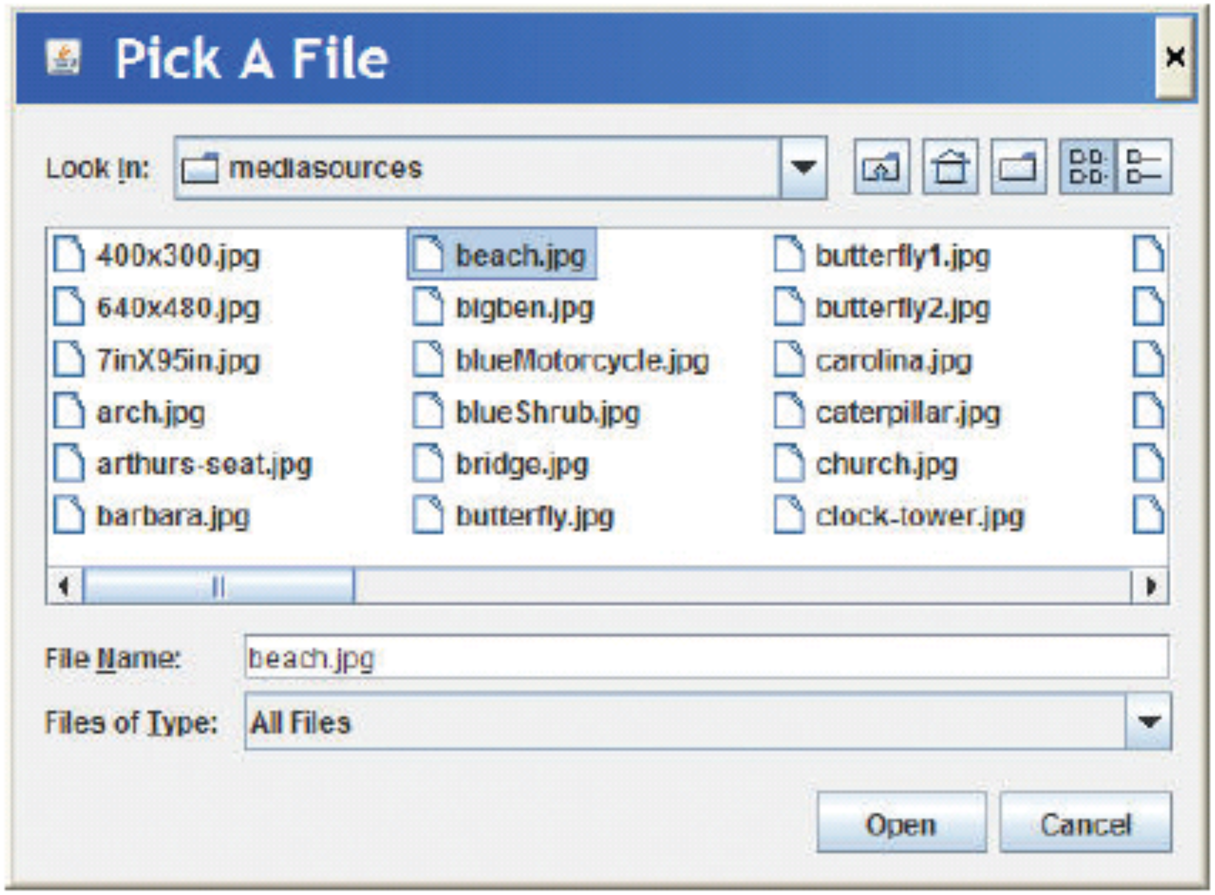
- Lab Exercise 0 & 1

Pictures

Picture Functions

- pickAFile()
  - Allows the user to pick a file
  - Takes no argument!

pickAFile() leads to… The File Picker! - UI


Pick A File

Look In: mediasources

400x300.jpg	beach.jpg	butterfly1.jpg
640x480.jpg	bigben.jpg	butterfly2.jpg
7inX95in.jpg	blueMotorcycle.jpg	carolina.jpg
arch.jpg	blueShrub.jpg	caterpillar.jpg
arthurs-seat.jpg	bridge.jpg	church.jpg
barbara.jpg	butterfly.jpg	clock-tower.jpg

File Name: beach.jpg

Files of Type: All Files

Picture Functions

- pickAFile()
- makePicture(filename)
  - creates and returns a picture object
- show(picture)
  - displays a picture in a window

Showing a picture

- Steps

1. Choose a file
2. Make it into a "picture"
3. Show the picture

```
myFile = pickAFile()
pic = makePicture(myFile)
show(pic)
```

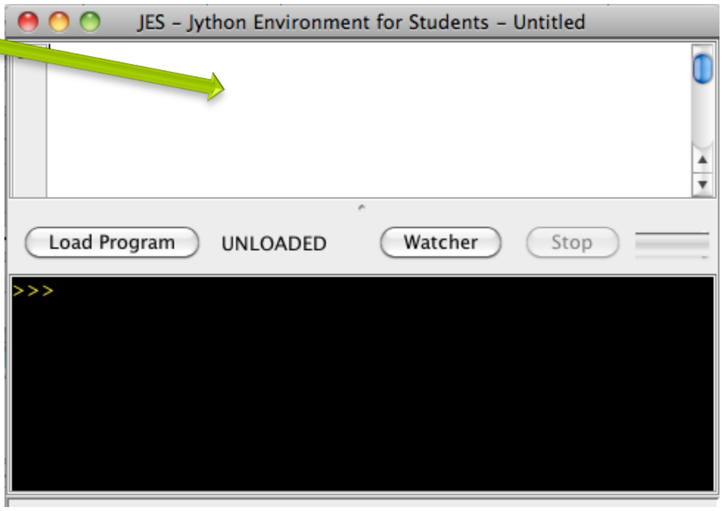
Alt: Nesting

```
show(makePicture(pickAFile()))
```

DEMO

Defining our own functions

```
def <name>(<arguments names>):
    return <expression>
```



```
>>>
```

- Functions:
  - function name
  - input values
  - Body

Defining our own functions

Structure of a function

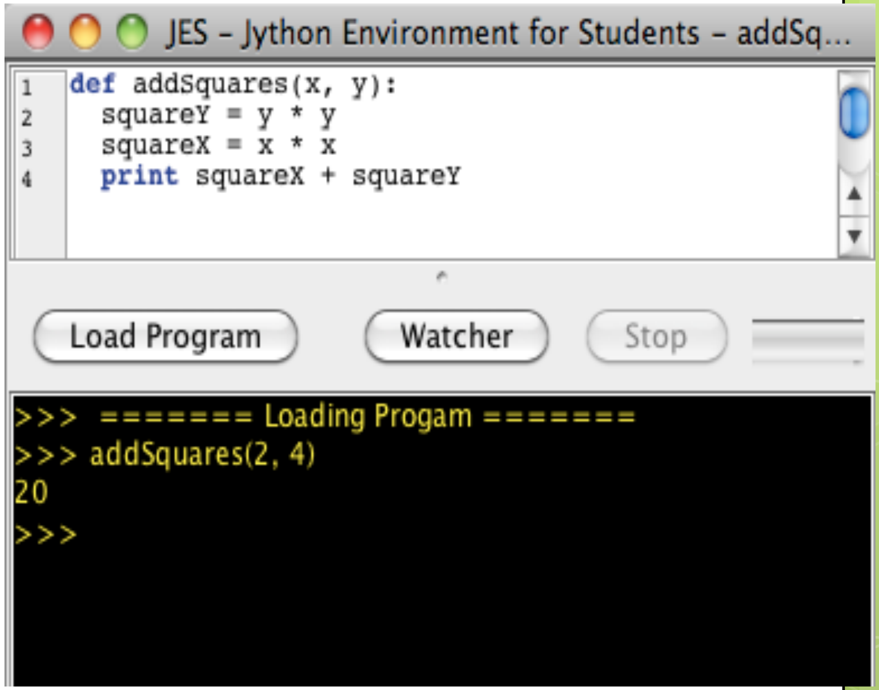
- **def**
- function name
- input values between parentheses
- colon
- **body** (indentation matters = 2 spaces)

```
def addSquares(x, y):
  squareX = x*x
  squareY = y*y
  return squareX + squareY
```

Nesting?

Blocking is indicated for you in JES

- Statements with same indentation = same block
- same block is enclosed in a blue box
- **DEMO**



```
def addSquares(x, y):
  squareY = y * y
  squareX = x * x
  print squareX + squareY
```

```
>>>  ======= Loading Progam =======
>>> addSquares(2, 4)
20
>>>
```

Try it yourself

- Lab Exercise 2 & 3

- (~15-20 minutes)

Day 1 Part 2

- Lab Part 2

Pictures

- An encoding that represents an image
  - height and width
  - filename
  - Containing *window* if it’s opened

```
>>> pic = makePicture(myFile)
>>> print pic
Picture, filename /Users/guzdial/mediasources/
   barbara.jpg height 294 width 222
```

Pixels

- Pictures are a bunch of little dots = pixel
  - *color*
  - Location (graph like format)

Methods

getPixel(picture,x,y) - retrieves a single pixel: more later

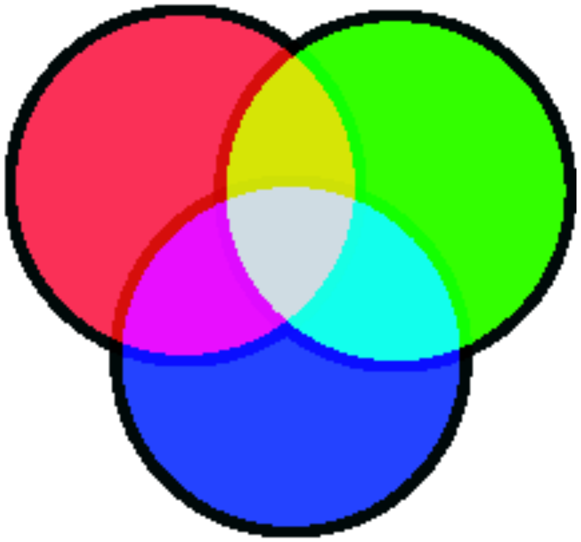
getPixels(picture) - gets *all* of them in a list

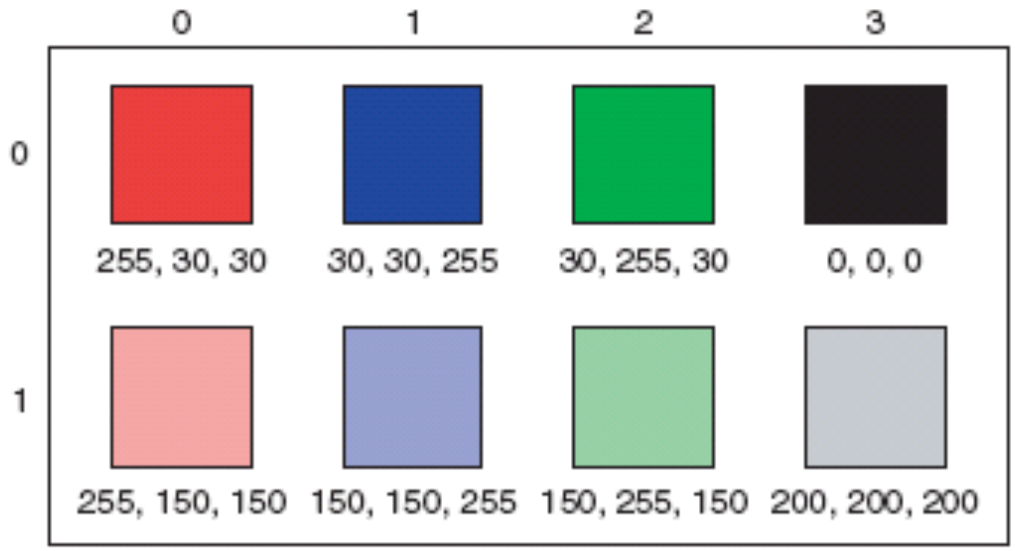
Example

```
>>> pixels=getPixels(pic)
>>> print pixels[0]
Pixel, color=color r=168 g=131 b=105
```

Colors: RGB

- In RGB, each color has three component colors:
  - Redness
  - Greenness
  - Blueness
- 0-255





	0	1	2	3
0	🟥 255, 30, 30	🟦 30, 30, 255	🟩 30, 255, 30	⬛ 0, 0, 0
1	🟥 255, 150, 150	🟦 150, 150, 255	🟩 150, 255, 150	⬜ 200, 200, 200

Pixel Methods

GETTERS

- ***Pixels***
  - *getRed(px)*   *getBlue(px)*   *getGreen(px)*
- ***Colors***
  - *getColor(px)*

SETTERS

- ***Pixels***
  - *setRed(px, val)*   *setBlue(px, val)*   …
- ***Color***
  - *setColor(px, col)*

We can change pixels directly…

```
>>> pict=makePicture(file)

>>> pix = getPixel(pict, 10, 100)

>>> setColor(pix, yellow)

>>> repaint(pict)
```

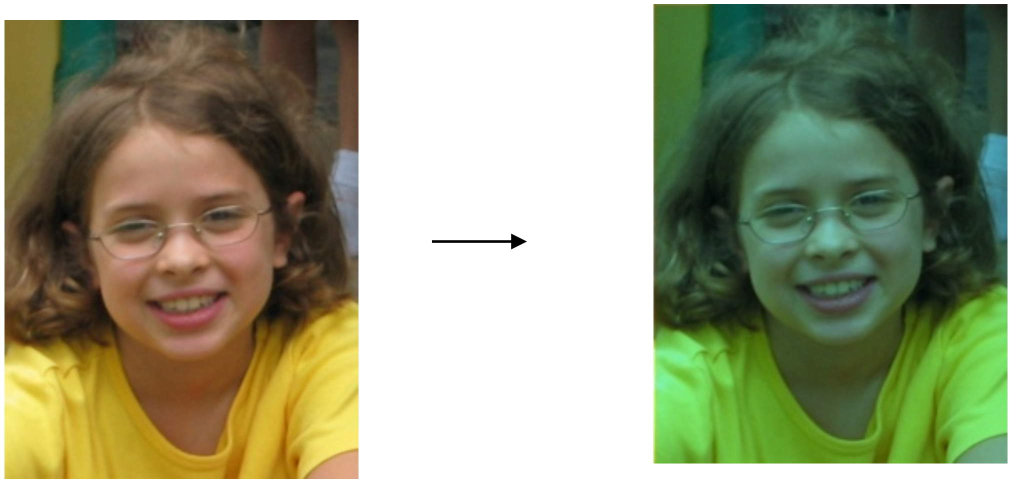
But that's *really* dull and boring to change each pixel at a time… Isn't there a better way?

How to change the entire picture!

decreaseRed()

```
def decreaseRed(picture):
```

decreases the red in all the pixels of a picture



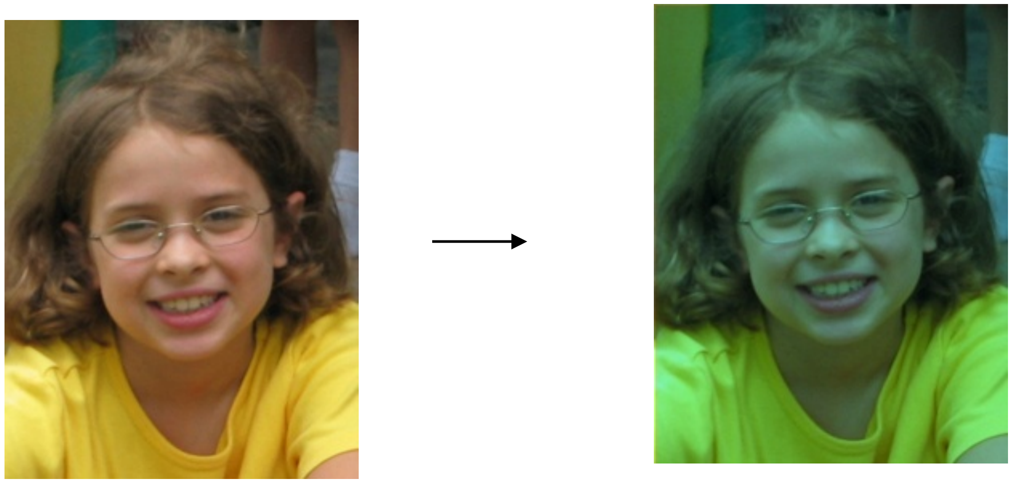
Decreasing the red in a picture

- **Recipe:** To decrease the red
- **Ingredients:** One picture, name it **pict**
- **Step 1:** Get <u>all</u> the pixels of **pict**. <u>For each</u> pixel **p** in the set of pixels…
- **Step 2:** Get the value of the red of pixel **p**, and set it to 50% of its original value

How to change the entire picture!

For loops!

```
def decreaseRed(picture):
  for each pixel in the picture
    get the red value of that pixel
    set the red value of that pixel to half the original
```



For loops

```
def decreaseRed(pict):
  allPixels = getPixels(pict)

  for pix in allPixels:
    value = getRed(pix)
    setRed(pix, value * 0.5)
```

The for loop

The body

- Note the indentation!